Application Program Interface Supplement
to the
Software Communications Architecture Specification

APPENDIX C

Generic Packet Building Block Service Definition

Revision Summary

1.0	Initial Release
2.2.1	Document numbering change for consistency with SCA main document numbering.
3.0	No change.

Table of Contents

List of Figures

List of Tables

C.1 INTRODUCTION.

C.1.1 Overview.

This document specifies SCA conformant building blocks that must be used by all API Service Definitions. The Packet Building Block (BB) provides a method to send user data between software resources and provides priority based flow control. The Packet Signals BB provides methods to signal the Service User that an event has occurred associated with the Packet BB flow control and is required only when using the packet BB. The Simple Packet BB provides a method to send user data between software resources and does not provide flow control. The Simple Packet BB should only be used when no flow control is required. The Signal Error BB provides a method for the Service Provider to signal the Service User of Errors. These Building Blocks support all protocols and waveforms and can be used at any of the interface locations as shown in Figure 1.

The service defined in this building block is used in conjunction with other layer building blocks to form a complete application-programming interface (API).

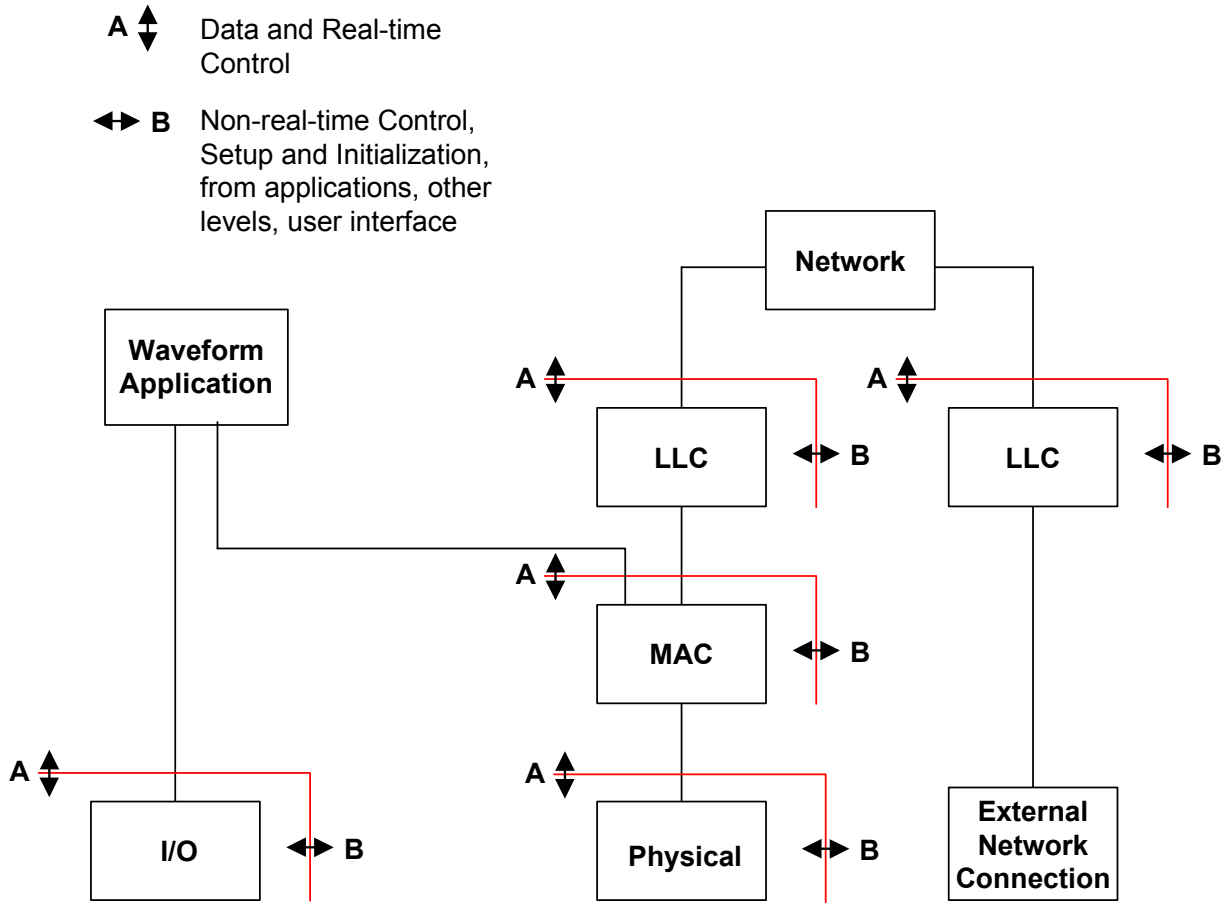

Figure 1. Service Definition Overview

C.1.2 Service Layer Description.

The Packet BB/Signal BB, Simple Packet BB, and Signal Error BB define the fundamental building blocks to provide an interface for software resources to communicate data to/from each other and signal errors. The Packet BB and Signal BB or Simple Packet BB shall be used by all APIs to 'push' data between software components (e.g., Physical, LLC, MAC, and I/O). The Packet BB/Signal BB provides for data transfer and flow control. The Simple Packet BB provides for data transfer and with no flow control. The Packet BB/Signal provides for two kinds of flow control: "Service User" polling of "Service Provider" for space available, or signals from the "Service Provider" to the "Service User" to indicate high-water, low-water and empty conditions of the queues. The Packet BB/Signal BB, Simple Packet BB, and Signal Error BB are parameterized and abstract, meaning that they only provide a template for creating a concrete class.

C.1.3 Modes of Service.

There are no specific modes. This document describes SCA building blocks for transfer of real time data with or without flow control. To form a complete service layer or application programming interface (API), these building blocks must be combined with other building blocks. When these building blocks are so used then it may be considered as constituting a data transfer service among other layer modes. This service is message-oriented and supports data transfer in self-contained units.

C.1.4 Service States.

The BB assumes no states.

C.1.5 Referenced Documents.

Programmer's Guide: STREAMS UNIX SYSTEM V, Release 4.2
JTRS-5000SCA, "Software Communication Architecture Specification (SCA)", 2.2.1, April 30, 2004.

C.2 UUID.

Not applicable.

C.3 SERVICES.

The features of the interface are defined in terms of the services provided by the Service Provider, and the individual primitives that may flow between the Service User and Service Provider.

The services are tabulated in Table 1 and described more fully in the remainder of this section.

Table 1. Cross-Reference of Services and Primitives

Service Group	Service	Primitives

Packet BB	Flow Control & QOS	readonly attribute unsigned short numOfPriorityQueues;
		
		short unsigned short spaceAvailable(in octet priorityQueueID);
		
		oneway void enableFlowControlSignals(in boolean enable);
		oneway void enableEmptySignal(: in boolean enable);
		
		oneway void setNumOfPriorityQueues(in octet) numOfPriorities);

Signal BB		oneway void signalHighWatermark (in short priorityQueueID); oneway void signalLowWatermark (in octet priorityQueueID); oneway void signalLowWaterMark (in octet priorityQueueID); oneway void signalEmpty();

Service Group	**Service**	**Primitives**
Packet BB	Data transfer	readonly attribute unsigned short minPayloadSize; readonly attribute unsigned short maxPacketSize; oneway void pushPacket (in octet priority, in ControlType control, in PayloadType payload);[**with flow control & QOS –uses Signal BB]** oneway void pushPacket (in ControlType control, in PayloadType payload););[**without flow control and QOS]** Note The instantiating API shall only use one format of pushPacket (with flow control & QOS or without flow control & QOS)
Error BB	Asynchronous error notification	oneway void signalError (in ErrorAugmentationType ErrorDetails);

C.3.1 Packet BB/Signal BB Data Transfer and Flow Control.

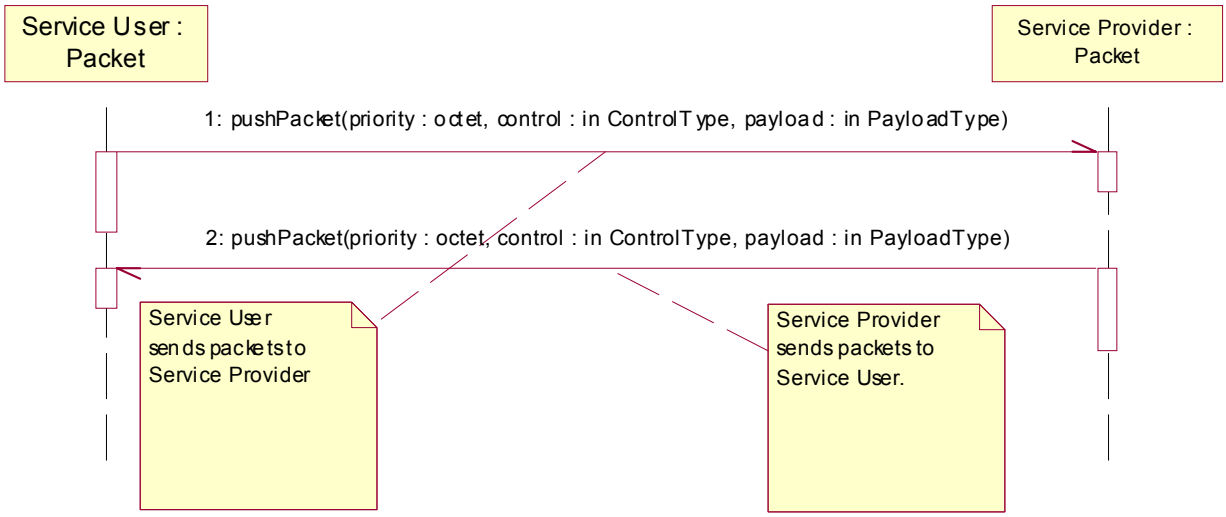

Figure 2. Packet BB/Signal BB data Transfer

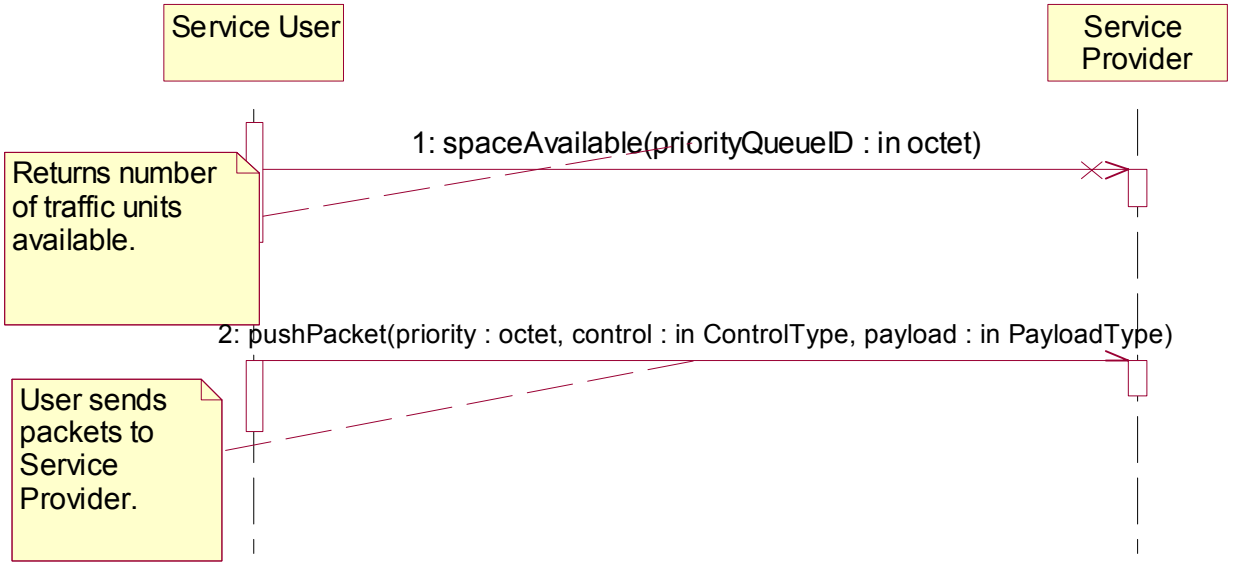

Figure 3. Packet BB/Signal BB Service User Flow Control

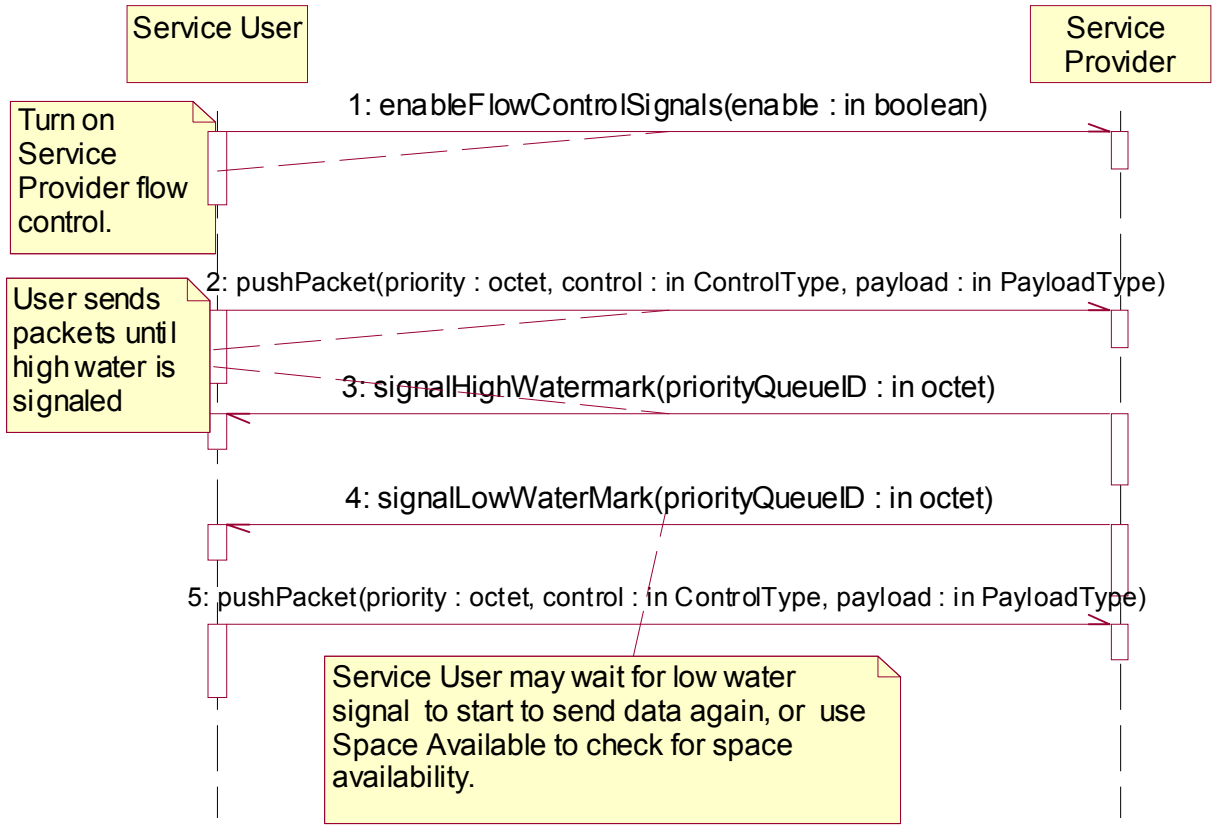

Figure 4. Packet BB/Signal BB Service Provider Flow Control

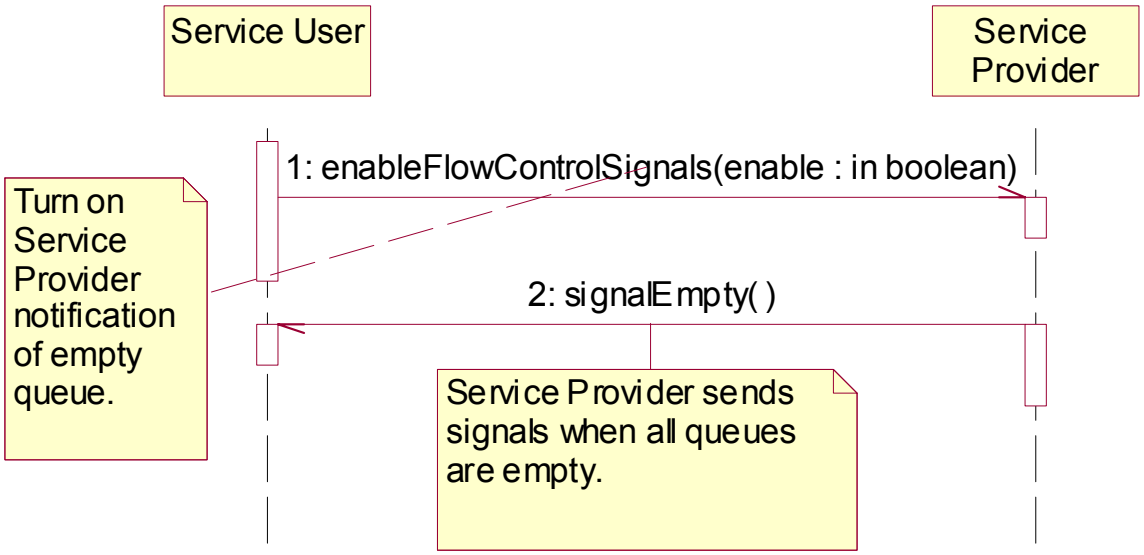

Figure 5. Packet BB/Signal BB Service Provider Empty Signal

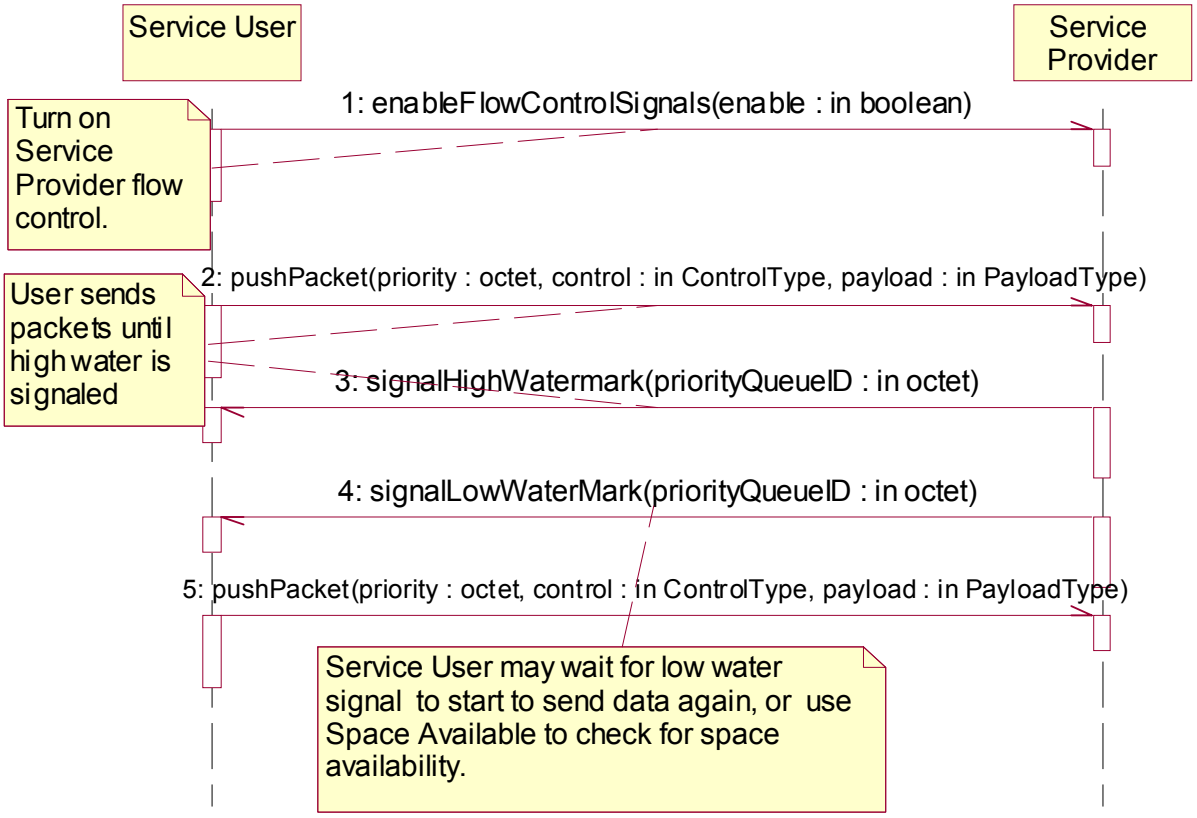

Figure 6. Packet BB/Signal BB Service Provider Flow Control Signals

C.3.2 Simple Packet Data Transfer.

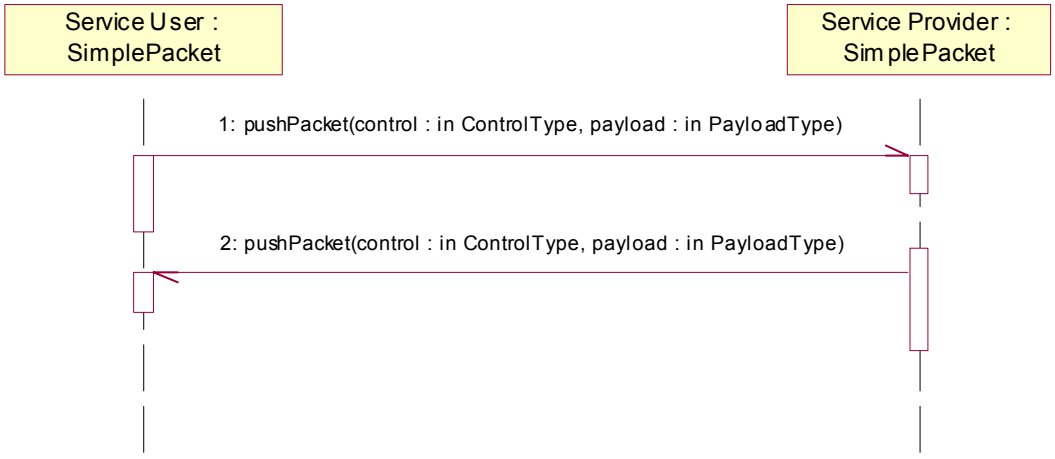

Figure 7. Simple Packet BB Data Transfer - No flow control

C.3.3 Error Signals.

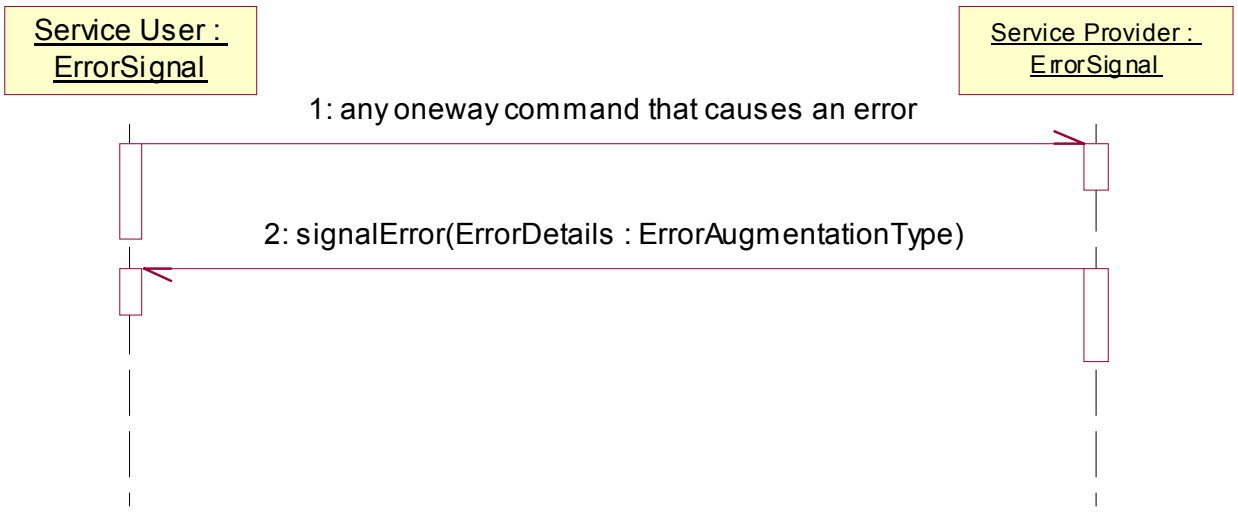

Figure 8. Error BB Asynchronous Error Signal

C.4 SERVICE PRIMITIVES.

C.4.1 Packet BB.

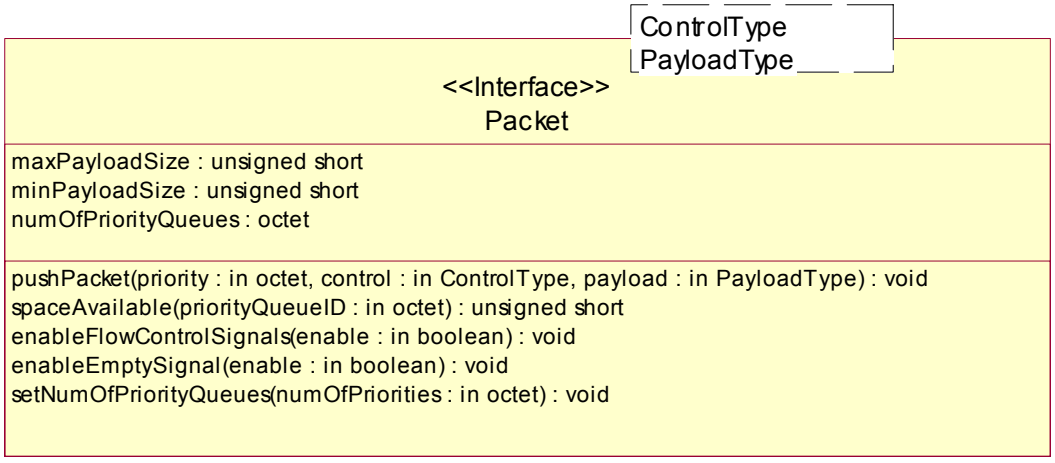

Figure 9. Packet BB

C.4.1.1 SpaceAvailable.

This operation provides the ability to poll the Service Provider to determine the amount of space available, in "element"s (see C.10.1.1 Parameters to be filled in by instantiating class for definition of "element"s), in the queue for the given priority. The Service User will poll the server provider to determine how much room is available in "element"s in the specified priority. When the operation is invoked, the server will respond with the amount of available space on the queue, in "element"s.

C.4.1.1.1 Synopsis.

unsigned short spaceAvailable(in octet priorityQueueID);

C.4.1.1.2 Parameters.

priorityQueueID : octet

This parameter indicates which PriorityQueue to check. The number of priority queues is set up via SetNumOfPriorityQueues primitive. If SetNumOfPriorityQueues has not been called, the default number of priority queues is 1.

C.4.1.1.3 State.

Any state.

C.4.1.1.4 New State.

Same state.

C.4.1.1.5 Response.

This operation responds with the amount of available space on the specified queue in "element"s.

C.4.1.1.6 Originator.

Service User.

C.4.1.1.7 Errors/Exceptions.

To be defined by instantiating API.

C.4.1.2 EnableFlowControlSignals.

This operation is used to activate and deactivate the 'water-mark' signals. The default is false (signals will not be generated).

C.4.1.2.1 Synopsis.

oneway void enableFlowControlSignal(: in boolean enable);

C.4.1.2.2 Parameters.

enable: boolean

false: The Service Provider will not generate signals to indicate the Lowwater and Highwater queue conditions. It is up to the Service User to poll the Service Provider to insure the Service Provider will not be starved or the queue will not overflow. The instantiating API should define behavior upon starvation or queue overflow.

true: The Service Provider will signal the Lowwater and Highwater queue conditions, to the Service User, when the Lowwater (queue near empty) and Highwater (queue near full) have been reached, respectively.

C.4.1.2.3 State.

NO_PROVIDER_WATERMARK_SIGNALS or PROVIDER_WATERMARK_SIGNALS

C.4.1.2.4 New State.

True -> PROVIDER_WATERMARK_SIGNALS

False-> NO_PROVIDER_WATERMARK_SIGNALS

C.4.1.2.5 Response.

None.

C.4.1.2.6 Originator.

Service User

C.4.1.2.7 Errors/Exceptions.

None.

C.4.1.3 EnableEmptySignal.

This operation is used to activate and deactivate the 'empty' signal. The signal will not be generated when set to False.

C.4.1.3.1 Synopsis.

oneway void enableEmptySignal(in boolean enable);

C.4.1.3.2 Parameters.

enable : boolean

false: The Service Provider will not generate a signal to indicate all queues are empty. It is up to the Service User to poll the Service Provider to insure the Service Provider will not be starved. The instantiating API should define behavior upon starvation.

true: The Service Provider will generate a signal to the Service User when the all queues are empty.

C.4.1.3.3 State.

NO_PROVIDER_EMPTY_SIGNAL or PROVIDER_ EMPTY_SIGNAL

C.4.1.3.4 New State.

mode(on) ->PROVIDER_EMPTY_SIGNAL

mode(off)-> NO_PROVIDER_EMPTY_SIGNAL

C.4.1.3.5 Response.

None.

C.4.1.3.6 Originator.

Service User

C.4.1.3.7 Errors/Exceptions.

None.

C.4.1.4 SetNumOfPriorityQueues.

This operation is used by the Service User to inform the Service Provider how many priority queues to provide.

C.4.1.4.1 Synopsis.

oneway void setNumOfPriorityQueues(in octet) numOfPriorities);

C.4.1.4.2 Parameters.

numOfPriorites : octet Specifies the number of priorities the Service Provider should process. (E.g., If the Service Provider set the value to 10, the Service User would send packets to the Service Provider with a priority of 0-9. Where 9 is the highest priority. Messages in priority 9 will be processed first by the Service Provider.)

C.4.1.4.3 State.

Any state.

C.4.1.4.4 New State.

Same state.

C.4.1.4.5 Response.

None.

C.4.1.4.6 Originator.

Service User.

C.4.1.4.7 Errors/Exceptions.

To be defined by instantiating API.

C.4.1.5 GetMaxPayLoadSize.

MaxPayloadSize indicates the maximum packet size in “element”s. This item allows the Service Provider to bound its internal memory usage. This operation is auto-generated from the associated attribute.

C.4.1.5.1 Synopsis.

readonly attribute unsigned short maxPacketSize;

C.4.1.5.2 Parameters.

None.

C.4.1.5.3 State.

Any state.

C.4.1.5.4 New State.

Same state.

C.4.1.5.5 Response.

Returns the maxPayLoadSize in “element”s.

C.4.1.5.6 Originator.

Service User.

C.4.1.5.7 Errors/Exceptions.

None.

C.4.1.6 GetMinPayLoadSize.

Indicates the maximum packet size in “element”s. This item allows the Service Provider to bound its internal memory usage. This operation is auto-generated from the associated attribute.

C.4.1.6.1 Synopsis.

readonly attribute unsigned short minPayloadSize;

C.4.1.6.2 Parameters.

None.

C.4.1.6.3 State.

Any state.

C.4.1.6.4 New State.

Same state.

C.4.1.6.5 Response.

This operation returns the minPayLoadSize in “element”s.

C.4.1.6.6 Originator.

Service User.

C.4.1.6.7 Errors/Exceptions.

None.

C.4.1.7 GetNumOfPriorityQueues

Provides the number of priority queues that the Service Provider is currently supporting. The default value is 1. The method SetNumOfPriorityQueues is used by the Service Provider to change the value. This operation is auto-generated from the associated attribute.

C.4.1.7.1 Synopsis.

readonly attribute octet numOfPriorityQueues

C.4.1.7.2 Parameters.

None.

C.4.1.7.3 State.

Any state.

C.4.1.7.4 New State.

Same state.

C.4.1.7.5 Response.

Returns the number of priority queues that are currently being provided by the Service Provider.

C.4.1.7.6 Originator.

Service User.

C.4.1.7.7 Errors/Exceptions.

None.

C.4.1.8 PushPacket.

"pushPacket" provides the ability of pushing data packets from the Service User to a Service Provider and from the Service Provider to the Service User. A packet is made up of two parts control and payload. The payload is queued according to the priority and is processed according to the information specified in control parameter.

C.4.1.8.1 Synopsis.

oneway void pushPacket (in octet priority, in ControlType control, in PayloadType payload);

C.4.1.8.2 Parameters.

priority: octet The priority queue to put the control and associated payload in. (See setNumOfPriorityQueues.)

control: ControlType: To be defined by instantiating API Service Definition

payload: PayloadType: To be defined by instantiating API Service Definition

C.4.1.8.3 State.

Any state.

C.4.1.8.4 New State.

Same state.

C.4.1.8.5 Response.

None.

C.4.1.8.6 Originator.

Service User.

C.4.1.8.7 Errors/Exceptions.

To be defined by instantiating API.

C.4.2 Signals BB.

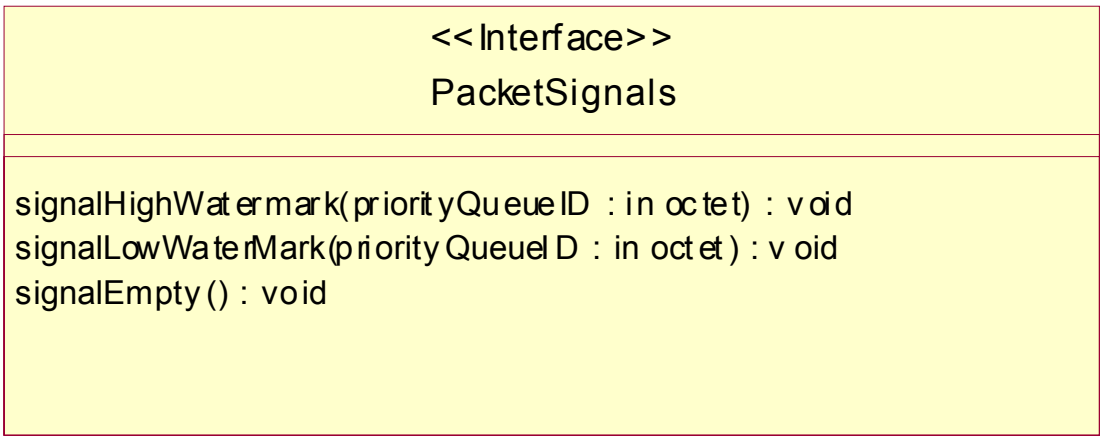

Figure 10. Signal BB

C.4.2.1 signal HighWatermark.

"*signalHighWaterMark*" provides the ability to signal the service user when a queue has reached the high water mark: the queue is full for the specified priority.

C.4.2.1.1 Synopsis.

oneway void signalHighWatermark(in octet priorityQueueID);

C.4.2.1.2 Parameters.

priorityQueueID : octet indicates the queue priority which has reached the high water mark. (See setNumOfPriorityQueues.)

C.4.2.1.3 State.

Any state.

C.4.2.1.4 New State.

Same state.

C.4.2.1.5 Originator.

Service Provider.

C.4.2.1.6 Errors/Exceptions.

None.

C.4.2.2 signal Low Watermark.

"*signalLowWaterMark*" provides the ability to signal the service user when a queue has reached the low water mark: the queue is near empty for the specified priority.

C.4.2.2.1 Synopsis.

oneway void signalLowWaterMark(in octet priorityQueueID);

C.4.2.2.2 Parameters.

priorityQueueID : octet indicates the queue priority which has reached the low water mark. (See setNumOfPriorityQueues.)

C.4.2.2.3 State.

Any state.

C.4.2.2.4 New State.

Same state.

C.4.2.2.5 Originator.

Service Provider.

C.4.2.2.6 Errors/Exceptions.

None.

C.4.2.3 signal Empty.

"*signalEmpty*" provides the ability to signal the service user when all priority queues are empty. (See setNumOfPriorityQueues.)

C.4.2.3.1 Synopsis.

oneway void signalEmpty();

C.4.2.3.2 Parameters.

None.

C.4.2.3.3 State.

Any state.

C.4.2.3.4 New State.

Same state.

C.4.2.3.5 Originator.

Service Provider

C.4.2.3.6 Errors/Exceptions.

None.

C.4.3 Simple Packet BB.

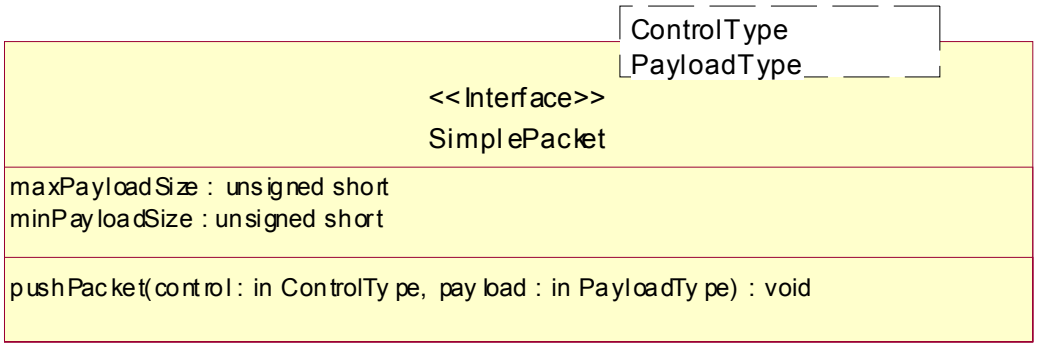

Figure 11. Simple Packet BB

C.4.3.1 PushPacket.

"pushPacket" provides the ability of pushing data packets from the Service User to a Service Provider and from the Service Provider to the Service User. A packet is made up of two parts control and payload. Simple Packet does not include any explicit flow control or queue handling mechanism.

C.4.3.1.1 Synopsis.

oneway void pushPacket (in ControlType control, in PayloadType payload);

C.4.3.1.2 Parameters.

control: ControlType: To be defined by instantiating API Service Definition

payload: PayloadType: To be defined by instantiating API Service Definition

C.4.3.1.3 State.

Any state.

C.4.3.1.4 New State.

Same state.

C.4.3.1.5 Response.

None.

C.4.3.1.6 Originator.

Service User.

C.4.3.1.7 Errors/Exceptions.

To be defined by instantiating API.

C.4.3.2 GetMaxPayLoadSize.

MaxPayloadSize indicates the maximum packet size in “element”s. This item allows the Service Provider to bound its internal memory usage This operation is auto-generated from the associated attribute.

C.4.3.2.1 Synopsis.

readonly attribute unsigned short maxPacketSize;

C.4.3.2.2 Parameters.

None.

C.4.3.2.3 State.

Any state.

C.4.3.2.4 New State.

Same state.

C.4.3.2.5 Response.

Returns the maxPayLoadSize in “element”s.

C.4.3.2.6 Originator.

Service User.

C.4.3.2.7 Errors/Exceptions.

None.

C.4.3.3 GetMinPayLoadSize.

Indicates the maximum packet size in “element”s. This item allows the Service Provider to bound its internal memory usage. This operation is auto-generated from the associated attribute.

C.4.3.3.1 Synopsis.

readonly attribute unsigned short minPayloadSize;

C.4.3.3.2 Parameters.

None.

C.4.3.3.3 State.

Any state.

C.4.3.3.4 New State.

Same state.

C.4.3.3.5 Response.

This operation returns the minPayLoadSize in “element”s.

C.4.3.3.6 Originator.

Service User.

C.4.3.3.7 Errors/Exceptions.

None.

C.4.4 Error Signal BB.

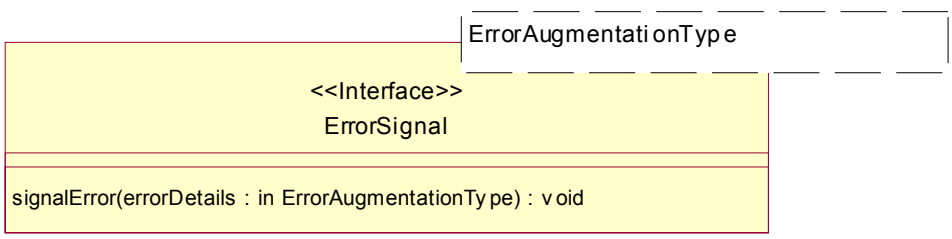

Figure 12. Error Signal BB

C.4.4.1 GetMaxPayLoadSize.

MaxPayloadSize indicates the maximum packet size in "element"s. This item allows the Service Provider to bound its internal memory usage This operation is auto-generated from the associated attribute.

C.4.4.1.1 Synopsis.

readonly attribute unsigned short maxPacketSize;

C.4.4.1.2 Parameters.

none

C.4.4.1.3 State.

Any state.

C.4.4.1.4 New State.

Same state.

C.4.4.1.5 Response.

Returns the maxPayLoadSize in "element"s.

C.4.4.1.6 Originator.

Service User.

C.4.4.1.7 Errors/Exceptions.

None.

C.4.4.2 GetMinPayLoadSize.

Indicates the maximum packet size in "element"s. This item allows the Service Provider to bound its internal memory usage. This operation is auto-generated from the associated attribute.

C.4.4.2.1 Synopsis.

readonly attribute unsigned short minPayloadSize;

C.4.4.2.2 Parameters.

None.

C.4.4.2.3 State.

Any state.

C.4.4.2.4 New State.

Same state.

C.4.4.2.5 Response.

This operation returns the minPayLoadSize in “element”s.

C.4.4.2.6 Originator.

Service User.

C.4.4.2.7 Errors/Exceptions.

None.

C.4.4.3 Signal Error.

This signal provides error information.

C.4.4.3.1 Synopsis.

oneway void signalError(in ErrorAugmentationType errorDetails);

C.4.4.3.2 Parameters.

errorDetails: ErrorAugmentationType: To be defined by instantiating API Service Definition.

C.4.4.3.3 State.

Any state.

C.4.4.3.4 New State.

Same state.

C.4.4.3.5 Response.

None.

C.4.4.3.6 Originator.

Service Provider

C.4.4.3.7 Errors/Exceptions.

None.

C.4.5 Common Structures.

The follow structures may be used by the instantiating API to create a *Control_Type*.

C.4.5.1 Time Structure

The time structure provides Service User and Service provider with a structure to communicate time information (e.g., when this packet should be transmitted). This requires the Service Provider and Service User to a have way to synchronize time between the Service Provider and Service User.

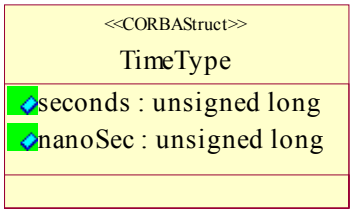

Figure 13. Time Structure

C.4.5.1.1 seconds.

Seconds in the future when an event should occur or in the past when an event has occurred. Seconds field is the seconds that have occurred since last synchronization epoch. (Epoch method shall be defined by instantiating API.) In most cases, the epoch will occur one time when the box is initialized.

C.4.5.1.2 NanoSec.

Nanosecond offset from the seconds field (described above) when a future event will occur or past event has occurred.

C.4.5.2 Stream Control Structure.

Stream control structure is used to control data groups sent between Service User and Service provider.

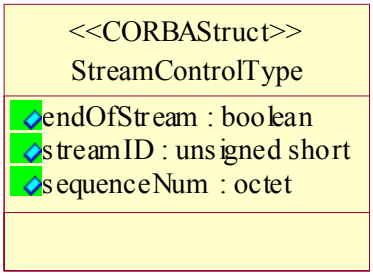

Figure 14. Stream Control

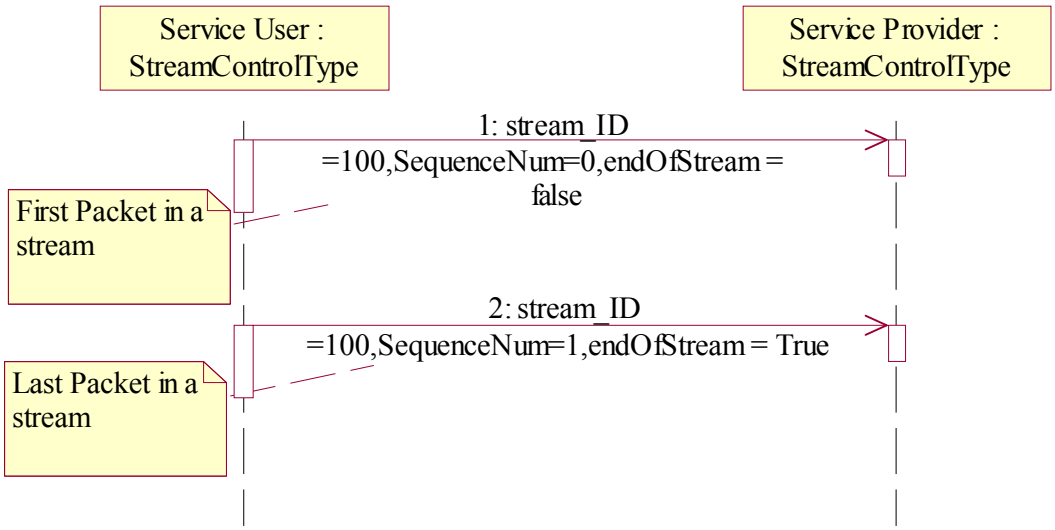

Figure 15. Stream Control Sequence Diagram

C.4.5.2.1 EndofStream.

Indicates last group of symbols for this hop: end of stream.

C.4.5.2.2 SteamId.

Identifies the groups of symbols to be transmitted or received in one hop.

C.4.5.2.3 SequenceNum.

Sequence number of the group of symbols within the stream sequence. The waveform application sets this value to zero at the start of stream. If value is set to zero, it indicates beginning of stream.

C.5 ALLOWABLE SEQUENCE OF SERVICE PRIMITIVES.

Table 2. High Water and Low Water and Empty On

Current State	Logical Event	Condition	Action	Next State
Queue normal	PushPacket	PushPacket does not cause queue to reach high water mark for the specified priority.	queue packet	Queue normal
		PushPacket causes queue to reach high water mark for the specified priority and there is room in the queue to put the Payload.	queue packet signalHighWatermark	Queue at high water mark
		PushPacket has Payload larger than can be put into the queue.	To be determined by instantiating API.	To be determined by instantiating API.
	packet extracted from queue	Packet extracted causes low water mark to be reached.	signalLowWatermark.	Queue at low water mark
		Packet extracted causes low water mark not to be reached		Queue normal
		Packet extracted causes all queues to be empty.	signalEmpty.	Queue empty
Queue at low water.	PushPacket	PushPacket does not cause the queue to exceed the low water mark for the specified priority.	queue packet	Queue at low water
		PushPacket causes queue to reach high water mark for the specified priority and there is room in the queue to put the Payload.	queue packet signalHighWatermark.	Queue at high water mark
		PushPacket causes the queue to exceed the low water mark for the specified priority but less than the high water mark.	queue packet	Queue normal
		PushPacket has a Payload larger than can be put into the queue	To be determined by instantiating API.	To be determined by instantiating API.

Table 2. High Water and Low Water and Empty On – Continued

Current State	Logical Event	Condition	Action	Next State
	packet extracted from queue	Packet extracted causes all queues to be empty.	signalEmpty.	Queue empty
		Packet extracted does not cause all queues to be empty.		Queue at low water
Queue at high water	PushPacket		To be determined by instantiating API.	To be determined by instantiating API.
	packet extracted from queue	Packet extracted causes low water mark to be reached.	signalLowWatermark.	Queue at low water mark
		Packet extracted causes all queues to be empty.	signalEmpty.	Queue empty
		Packet extracted causes the specified queue to be less than high water and greater than low water.		Queue normal
		Queue is still greater than or equal to high water mark.		Queue at high water
Queue empty	PushPacket	PushPacket causes the queue to be greater than the low water mark for the specified priority but less than the high water mark.	queue packet	Queue normal
		PushPacket causes the queue to equal the low water mark for the specified priority.	signalLowWatermark.	Queue at low water mark
		PushPacket causes queue to reach high water mark for the specified priority and there is room in the queue to put the Payload.	queue packet signalHighWatermark.	Queue at high water mark
		PushPacket has a Payload larger than can be put into the queue.	To be determined by instantiating API.	To be determined by instantiating API.
	attempt to extract packet from queue	To be determined by instantiating API.	To be determined by instantiating API.	To be determined by instantiating API.

Table 3. High Water and Low Water Off and Empty On

Current State	Logical Event	Condition	Action	Next State
Queue normal	PushPacket	PushPacket does not cause queue to reach high water mark for the specified priority.	queue packet	Queue normal
		PushPacket causes queue to reach high water mark for the specified priority and there is room in the queue to put the Payload.	queue packet	Queue at high water mark
		PushPacket has Payload larger than can be put into the queue.	To be determined by instantiating API.	To be determined by instantiating API.
	packet extracted from queue	Packet extracted causes low water mark to be reached.		Queue at low water mark
		Packet extracted causes low water mark not to be reached.		Queue normal
		Packet extracted causes all queues to be empty.	signalEmpty.	Queue empty
Queue at low water.	PushPacket	PushPacket does not cause the queue to exceed the low water mark for the specified priority.	queue packet	Queue at low water
		PushPacket causes queue to reach high water mark for the specified priority and there is room in the queue to put the Payload.	queue packet	Queue at high water mark
		PushPacket causes the queue to exceed the low water mark for the specified priority but less than the high water mark.	queue packet	Queue normal
		PushPacket has a Payload larger than can be put into the queue	To be determined by instantiating API.	To be determined by instantiating API.

Table 3. High Water and Low Water Off and Empty On - Continued

Current State	Logical Event	Condition	Action	Next State
	packet extracted from queue	Packet extracted causes all queues to be empty.	signalEmpty.	Queue empty
		Packet extracted does not cause all queues to be empty.		Queue at low water
Queue at high water	PushPacket		To be determined by instantiating API.	To be determined by instantiating API.
	packet extracted from queue	Packet extracted causes low water mark to be reached.		Queue at low water mark
		Packet extracted causes all queues to be empty.	signalEmpty.	Queue empty
		Packet extracted causes the specified queue to be less than high water and greater than low water.		Queue normal
		Queue is still greater than or equal to high water mark		Queue at high water
Queue empty	PushPacket	PushPacket causes the queue to be greater than the low water mark for the specified priority but less than the high water mark.	queue packet	Queue normal
		PushPacket causes the queue to equal the low water mark for the specified priority.		Queue at low water mark
		PushPacket causes queue to reach high water mark for the specified priority and there is room in the queue to put the Payload	queue packet	Queue at high water mark
		PushPacket has a Payload larger than can be put into the queue.	To be determined by instantiating API.	To be determined by instantiating API.
	attempt to extract packet from queue	To be determined by instantiating API.	To be determined by instantiating API.	To be determined by instantiating API.

Table 4. High Water and Low Water and Empty Off

Current State	Logical Event	Condition	Action	Next State
Queue normal	PushPacket	PushPacket does not cause queue to reach high water mark for the specified priority.	queue packet	Queue normal
		PushPacket causes queue to reach high water mark for the specified priority and there is room in the queue to put the Payload.	queue packet	Queue at high water mark
		PushPacket has Payload larger than can be put into the queue.	To be determined by instantiating API.	To be determined by instantiating API.
	packet extracted from queue	Packet extracted causes low water mark to be reached.		Queue at low water mark
		Packet extracted causes low water mark not to be reached.		Queue normal
		Packet extracted causes all queues to be empty.		Queue empty
Queue at low water.	PushPacket	PushPacket does not cause the queue to exceed the low water mark for the specified priority.	queue packet	Queue at low water
		PushPacket causes queue to reach high water mark for the specified priority and there is room in the queue to put the Payload.	queue packet	Queue at high water mark
		PushPacket causes the queue to exceed the low water mark for the specified priority but less than the high water mark.	queue packet	Queue normal
		PushPacket has a Payload larger than can be put into the queue.	To be determined by instantiating API.	To be determined by instantiating API.

Table 4. High Water and Low Water and Empty Off - Continued

Current State	Logical Event	Condition	Action	Next State
	packet extracted from queue	Packet extracted causes all queues to be empty.		Queue empty
		Packet extracted does not cause all queues to be empty.		Queue at low water
Queue at high water	PushPacket		To be determined by instantiating API.	To be determined by instantiating API.
	packet extracted from queue	Packet extracted causes low water mark to be reached.		Queue at low water mark
		Packet extracted causes all queues to be empty.		Queue empty
		Packet extracted causes the specified queue to be less than high water and greater than low water.		Queue normal
		Queue is still greater than or equal to high water mark.		Queue at high water
Queue empty	PushPacket	PushPacket causes the queue to be greater than the low water mark for the specified priority but less than the high water mark.	queue packet	Queue normal
		PushPacket causes the queue to equal the low water mark for the specified priority.		Queue at low water mark
		PushPacket causes queue to reach high water mark for the specified priority and there is room in the queue to put the Payload.	queue packet	Queue at high water mark
		PushPacket has a Payload larger than can be put into the queue.	To be determined by instantiating API.	To be determined by instantiating API.
	attempt to extract packet from queue	To be determined by instantiating API.	To be determined by instantiating API.	To be determined by instantiating API.

C.6 PRECEDENCE OF SERVICE PRIMITIVES.

Not Applicable.

C.7 SERVICE USER GUIDELINES.

Not Applicable.

C.8 SERVICE PROVIDER-SPECIFIC INFORMATION.

Not Applicable.

C.9 IDL.

The IDL for an API is generated using the parameterized Classes of the building blocks to generate concrete classes with Waveform specific types and attributes. The Building Block documented herein is not intended to be instantiated directly in a UML diagram. The parameterized classes define the attributes and data types that are unique for each waveform.

The parameterized class is a template from with to generate user specific API definitions. To generate valid IDL from this Building Block, a concrete class must generated that replaces the parameterized items with the user specific types and attributes. The completed UML diagram will not contain any reference to parameterized classes. Only concrete classes that were derived from them.

C.10 UML.

C.10.1 Packet & Signal UML model.

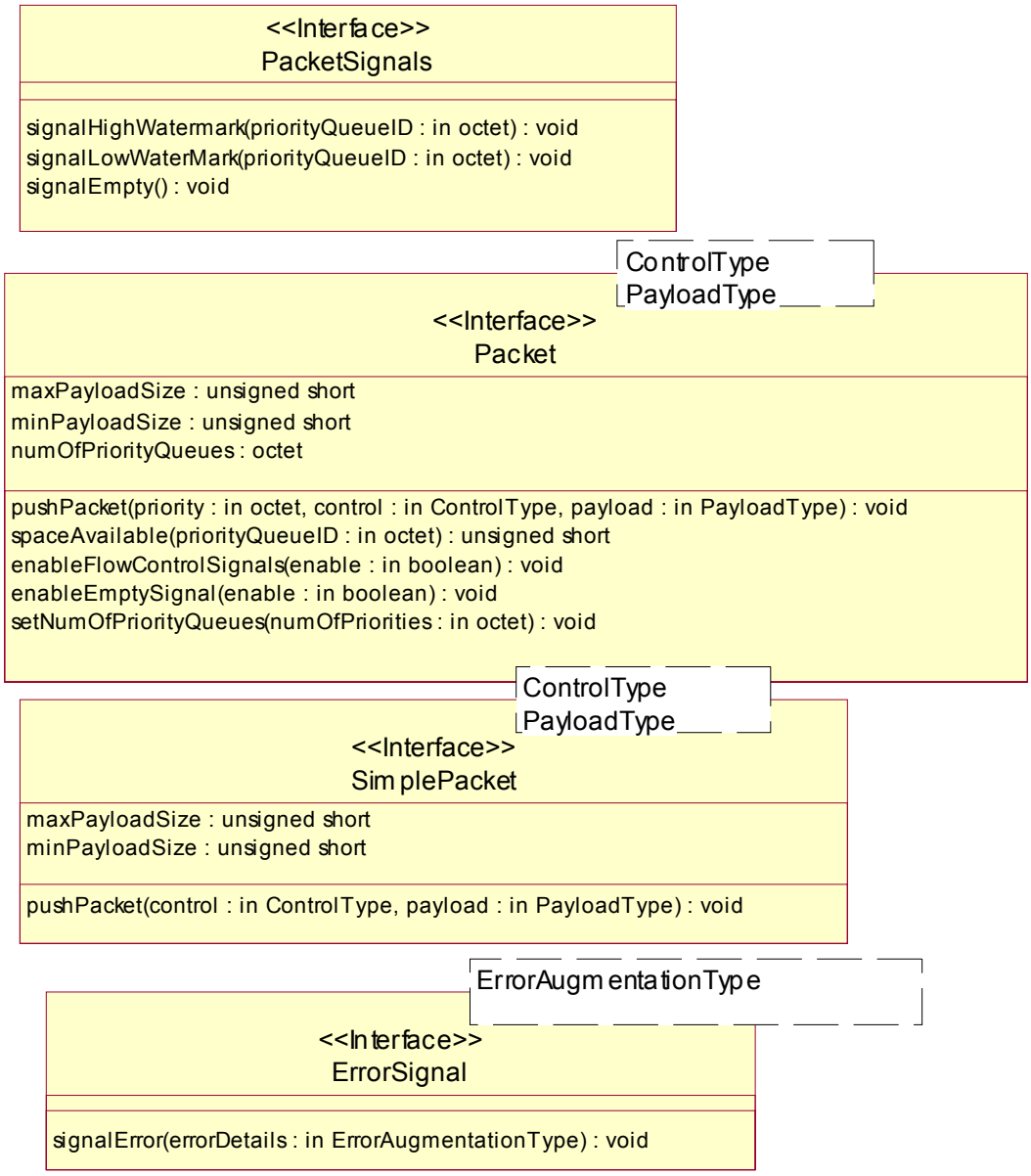

Figure 16. UML Diagram

C.10.1.1 Parameters to be filled in by instantiating class.

C.10.1.1.1 Payload_Type.

The instantiating API defines this type. "Payload_Type"can be defined to be a single "element" or an array of "element"s. The "element" is user defined for each Service User/Provider pair. Typical examples of "element"s are Octets, Audio Samples, digital waveform data words, etc. For example, a waveform that has an Audio I/O API, LLC API, Mac API, and a Physical API. The Payload_Type would be defined specifically for the needs of the APIs. The I/O API could define the "element" to be a 12-bit sample, and define Payload_Type to be a sequence of unsigned shorts (bits 0-11 are the samples). The LLC API and Mac API would probably use an Octet as the "element" and define the "Payload_Type" be an array of Octets. For the Physical API down-stream (to antenna) *pushpacket* instantiation, the "element" could be a 10-bit sample; and Payload_Type define to be a sequence of unsigned shorts (bits 0-9 are the samples). For Physical API up-stream (from the antenna) *pushpacket* instantiation, the "element" could be a structure that contained data plus soft decision information for each sample

C.10.1.1.2 Control_Type.

The instantiating API defines this type. This is the control information associated with *PayLoad_Type*. Examples of Control Type information are Begin of Stream, End of Stream, Next Frequency, Time of Transmission, Quality of Service (QOS) e.g., – bypass flow control.

C.10.1.1.3 ErrorAugmentationType

The instantiating API defines this type. This is the information associated with errors generated by the Service Provider. Examples of ErrorAugmentationType information are structures and enumeration types (memory capacity exceeded, invalid queue specified, etc)